Hoiio SDK for Python

Release 0.1.5

Junda Ong

December 24, 2014

Contents

This is an unofficial Python SDK to consume Hoiio API telephony services.

Using the SDK, developers can easily invoke any of Hoiio RESTful APIs. Start with *Quick Start*, and then proceed to *Complete Guide* that includes all the magic that you need to know!

CHAPTER 1

Contents

1.1 Quick Start

Install Hoiio SDK from Python Cheeseshop.

```
sudo pip install hoiio
```

The SDK will be installed, together with it's dependencies. There is only 1 dependency:

- Requests

You do NOT need to install the dependencies since they will be automatically installed with *pip install*.

Start by initializing the service with your Hoiio credentials (a pair of App ID and Access Token). If you have not, register a developer account from Hoiio and create an app from the portal.

Then make Hoiio API requests like this:

```
from hoiio import Hoiio

# REPLACE WITH YOUR HOIIO CREDENTIALS
Hoiio.init('MY_APP_ID', 'MY_ACCESS_TOKEN')

# Makes a voice call back
Hoiio.voice.call('+6511111111', '+6522222222')

# Send an SMS
Hoiio.sms.send('+6511111111', 'Hoiio World~')

# Fax a document
Hoiio.fax.send('+6511111111', '/path/to/my/file.pdf')

# IVR sequence
Hoiio.ivr.dial('+6511111111')
# After the call is picked up..
Hoiio.ivr.gather(session, msg = 'For sales enquiry, please press 1. For technical support, please pre
# Finally, hangup the call
Hoiio.ivr.hangup(session)
```

This is *merely* a quick start guide.

Refer to *Complete Guide* for detailed usage.

1.2 Complete Guide

Go through the complete guide, which shows you all the functionalities that the SDK provides.

1.2.1 Basics

Setting Up

Before you begin, make sure you have already installed the SDK. Refer to *Quick Start* if you have not. Basically you do a *pip install hoiio*.

To use the package, you import `Hoiio` from `hoiio`.

Then initialize the service with your Hoiio credentials (a pair of App ID and Access Token). You can create an app and get the Hoiio credentials by logging in at http://developer.hoiio.com.

```
from hoiio import Hoiio

Hoiio.init('MY_APP_ID', 'MY_ACCESS_TOKEN')
```

We use a singleton pattern [1] to init the credentials, so you only need to call *init(...)* once.

Note: Obviously, you need to replace MY_APP_ID and MY_ACCESS_TOKEN with YOUR Hoiio credentials. Obtain them from http://developer.hoiio.com.

Note: We use a singleton pattern because most of the time, you are using the same set of credentials for your application. However, if you need to use different credentials simultaneously, you can override this behaviour by providing the keys *app_id* and *access_token* as optional keyword arguments.

Requests

Hoiio SDK is designed to let you write less code.

It is designed to be as simple as possible, but no simpler [2].

Hence, it has minimal number of classes which you need to know. The most important is `Hoiio`, which you can use to make various API requests, such as:

- Hoiio.voice.call(...)
- Hoiio.sms.send(...)
- Hoiio.fax.send(...)
- Hoiio.ivr.dial(...)
- Hoiio.ivr.gather(...)

As you can see, the services are categorized as voice, sms, fax, ivr, number and account. This correspond to what Hoiio API provides.

The **required parameters** are passed in as **positional arguments** to the methods.

The **optional parameters** are passed in as **keyword arguments** after the positional arguments.

[1] http://en.wikipedia.org/wiki/Singleton_pattern
[2] http://en.wikiquote.org/wiki/Albert_Einstein

Example of a mix of required and optional parameters:

```
res = Hoiio.CATEGORY.REQUEST(REQUIRED_1, REQUIRED_2, key1=OPTIONAL_1)
```

You will see many of the APIs are in the same pattern.

Response

After you make a service request, a `Response` object will be returned.

You can access the fields of the response object. It corresponds to the response parameters in Hoiio API.

```
print res.txn_ref
# 'TX-1234'
```

You can also access additional fields.

```
# The exact http response body
print res.text
# '{"txn_ref": "AA-C-3070102","status": "success_ok"}'

# The http response body in JSON
print res.json
# {'txn_ref': 'AA-C-3070102', 'status': 'success_ok'}
```

Lastly, the SDK uses Requests, a *HTTP python package for Humans*.

You can access the Response object, which gives you access to fields like the HTTP headers and status code.

```
print res.response.headers
# {'content-length': '56170', 'x-content-type-options': 'nosniff', 'x-cache':
'HIT from cp1006.eqiad.wmnet, MISS from cp1010.eqiad.wmnet', 'content-encoding':
'gzip', 'age': '3080', 'content-language': 'en', 'vary': 'Accept-Encoding,Cookie',
'server': 'Apache', 'last-modified': 'Wed, 13 Jun 2012 01:33:50 GMT',
'connection': 'close', 'cache-control': 'private, s-maxage=0, max-age=0,
must-revalidate', 'date': 'Thu, 14 Jun 2012 12:59:39 GMT', 'content-type':
'text/html; charset=UTF-8', 'x-cache-lookup': 'HIT from cp1006.eqiad.wmnet:3128,
MISS from cp1010.eqiad.wmnet:80'}
```

Note: There are 2 different objects with the same name here. Hoiio returns a response (*res*). Within this *res*, there is a *response* object (*res.response*) provided by Requests. You usually need not deal with *res.response*, unless you want to access the HTTP raw data.

You can even access the request headers, if you need it.

```
print res.response.request.headers
# {'Accept-Encoding': 'identity, deflate, compress, gzip',
'Accept': '*/*', 'User-Agent': 'python-requests/0.13.1'}
```

1.2.2 Voice

Make call back

Makes a voice call back between 2 phone numbers.

```
res = Hoiio.voice.call('+6511111111', '+6522222222')
```

There are some optional parameters that you can use as documented in Hoiio Voice API. These optional parameters can be passed in as keyword arguments.

```
res = Hoiio.voice.call('+6511111111', '+6522222222',
    caller_id = 'private',
    max_duration = 600,
    tag = 'myapp',
    notify_url = 'http://my.server.com/myscript'
)
```

The example above will call with a private caller ID (shown only to dest2), the call will be cut off in 10 minutes (600 sec), tagged with 'myapp' for the application reference, and the server's script will be notified on the call status.

We have shown how you can make an API request. But what about handling the response?

Hoiio API response can be accessed as fields of a Response object.

```
res = Hoiio.voice.call('+6511111111', '+6522222222')

print res.txn_ref
# 'AA-S-141147'

print res.is_success()
# True
```

One of the most important field is *txn_ref* - a transaction for the API. All chargeable API (eg. making a call back, sending an SMS) will return 1 or more *txn_ref*.

It is also possible for the API to return unsuccessful. The response status will then return the error code eg. *error_invalid_http_method*, *error_insufficient_credit*, *error_rate_limit_exceeded*, etc. If *is_success()* returns *False*, then there is an error, and you might want to log the error.

```
res = Hoiio.voice.call('+6511111111', '+6522222222')

if res.is_success():
    print 'The txn ref:', res.txn_ref
else:
    # The API failed. Print the error code.
    print 'Error:', res.status
```

Make conference call

Voice call back connects only 2 phone numbers. As an extension, voice conference can connect multiple phone numbers (currently limited to 8 per room).

```
# Call 3 phones and put them in a conference
res = Hoiio.voice.conference('+6511111111', '+6522222222', '+6533333333')

# Find out the room id
print res.room
# 'MY-ROOM'

# The transaction reference ID correspond to each of the phone number
print res.txn_refs
# ['TX-1', 'TX-2', 'TX-3']
```

It is not neccessary that you supply all the phone numbers in 1 API request. You could at some point in time add another participant into an existing conference room.

```
# Add another participant into the conference room
res = Hoiio.voice.conference('+6544444444', room='MY-ROOM')

print res.txn_refs
# ['TX-4']
```

Hangup call

You may also at any point in time hangup any of the participant eg. kick him out of the conference room. In the example below, 'TX-4' refers to the *txn_ref* of +6544444444.

```
# Hangup one of the phone.
res = Hoiio.voice.hangup('TX-4')
```

Hangup is applicable to both conference call and call back. The difference is that a call back is considered 1 transaction, so hangup will disconnect both the phones, whereas a conference call is made up of multiple participants (each with their own transaction), so hangup will disconnect the participants individually.

Retrieve call status

You can find out the call status of a particular transaction.

```
res = Hoiio.voice.status('TX-1234')

print res.txn_ref
# 'TX-1234'

print res.tag
# 'my-tag'

print res.date
# datetime.datetime(2012, 1, 31, 12, 6, 15)

print res.dest1
# '+6511111111'

print res.dest2
# '+6522222222'

print res.call_status_dest1
# 'answered'

print res.call_status_dest2
# 'answered'

print res.duration
# 2

print res.currency
# 'SGD'

print res.rate
# 0.018

print res.debit
# 0.036
```

There are many information you can get from a call status. Most of the fields are returned as string or int or float. For 'date', a python datetime is returned.

Note: All datetime is in GMT+8.

The Call Status can also be used to query for the live status of a call eg. is it still ongoing?

```
res = Hoiio.voice.status('TX-1234')

print res.call_status_dest1
# 'ongoing'
```

Retrieve call history

Query for all the transactions.

```
res = Hoiio.voice.history()

print res.total_entries_count
# 234

print res.entries_count
# 100

for entry in res.entries:
    print entry.txn_ref
    print entry.date
    # etc ..
```

Each entry has similar fields to that of Call Status.

The query history API will fetch the transationcs in batches of 100. To go to the next page:

```
res = Hoiio.voice.history(page=2)
```

Retrieve call rate

You could find out how much the call back will cost before you actually make the call.

```
res = Hoiio.voice.rate('+6511111111', '+6522222222')

print res.currency
# 'SGD'

print  res.rate
# 0.036

print res.talktime
# 2352
```

The about call will cost $0.036 (Singapore Dollars), and with the account credit balance, the call can last 2352 minutes.

If you don't want to use API to find out the cost, you could refer to the Pricing Page.

1.2.3 SMS

Send an SMS

Send a single SMS to a phone. What's better than a Hoiio World message.

```
res = Hoiio.sms.send('Hoiio World', '+6511111111')
```

There are some optional parameters that you can use as documented in Hoiio SMS API. These parameters can be passed in as keyword arguments in all of the methods.

```
res = Hoiio.sms.send('Hoiio World', '+6511111111',
    sender_name = 'Citibank',
    tag = 'myapp',
    notify_url = 'http://my.server.com/myscript'
)
```

One of the very useful feature is `sender_name`. It should only be used if you have activated the SMS Rebranding feature. You can change to any alphanumeric eg. a company name or other phone numbers.

The `notify_url` should be your web server. Hoiio will post notification to this URL on the SMS delivery status. You need this to know if the SMS is "queued" of "delivered".

Hoiio API response can be accessed as fields of a `Response` object.

```
res = Hoiio.sms.send('Hoiio World', '+6511111111')

print res.txn_ref
# 'AA-S-1234'

print res.is_success()
# True
```

The `txn_ref` is a very important field - a transaction for the API. All chargeable API (eg. making a call back, sending an SMS) will return 1 or more txn_ref.

Send bulk SMS

Bulk SMS API is a convenient extension to Send an SMS. It sends up to 1,000 SMS in a single request. This is useful if you want to send the same message to multiple phone numbers.

The phone numbers are passed in as variable arguments to the method.

```
# SMS to 2 numbers (up to 1000 numbers)
res = Hoiio.sms.bulk_send('Hoiio World', '+6511111111', '+652222222',
    notify_url = 'http://my.server.com/myscript'
)

# The bulk_txn_ref that encapsulate individual txn_ref
print res.bulk_txn_ref
# 'AA-B-1234'
```

It is recommended that you make use of `notify_url` to track the status of the individual SMS. The `txn_ref` of the individual SMS will be provided during the notification phase.

Receive SMS

Hoiio supports receiving SMS. Developers need to purchase an SMS enabled number from Hoiio. At the point of writing (Sep 2012), the only country that has SMS enabled number is the US.

To get a notification from Hoiio whenever you receive an SMS at the number, you will need to go to Hoiio's developer portal and configure the Notify URL, or use the *Number API* to configure.

Retrieve SMS status

You can find out the SMS status of a particular transaction.

There are many information you can get from a SMS status. Most of the fields are returned as string, int or float. For 'date', a python datetime is returned. Note the datetime is in GMT+8.

```
res = Hoiio.sms.status('TX-1234')

print res.txn_ref
# 'TX-1234'

print res.sms_status
# 'delivered'

print res.dest
# '+6511111111'

print res.date
# datetime.datetime(2012, 1, 31, 12, 6, 15)

print res.tag
# 'my-tag'

print res.split_count
# 2

print res.currency
# 'SGD'

print res.rate
# 0.032

print res.debit
# 0.064
```

Retrieve SMS history

Query for all the transactions.

```
res = Hoiio.sms.history()

print res.total_entries_count
# 234

print res.entries_count
# 100

for entry in res.entries:
```

```
    print entry.txn_ref
    print entry.date
    # etc ..
```

Each entry has similar fields to that of SMS Status (see Retrieve SMS status).

The query history API will fetch the transationcs in batches of 100. To go to the next page:

```
res = Hoiio.voice.history(page=2)
```

You can also filter by dates. The date format is 'YYYY-MM-DD HH:MM:SS' (GMT+8).

```
res = Hoiio.voice.history(from='2012-01-01 08:00:00', to='2012-12-31 08:00:00')
```

Retrieve SMS rate

Find out how much an SMS will cost before you actually send it.

```
res = Hoiio.sms.rate('Hoiio World', '+6511111111')

print res.currency
# 'SGD'

print res.rate
# 0.032

res.split_count
# 2

res.total_cost
# 0.064

res.is_unicode
# False
```

You can also find out how much it cost to receive an SMS on your Hoiio number. Hoiio supports receiving SMS for only a few countries. In the example below, you own the Hoiio number +6599999999.

```
res = Hoiio.sms.rate_in('+6599999999')

print res.currency
# 'SGD'

print res.rate
# 0.01
```

Note that Hoiio charges per incoming SMS, regardless of the message size or unicode.

If you don't need to use API to find out the cost (it seldom change anyway), you could refer to Hoiio Pricing Page.

1.2.4 IVR

The IVR APIs are grouped into :

1. Start blocks: Answer, Dial
2. Middle blocks: Play, Gather, Record, Monitor
3. End blocks: Hangup, Transfer

In a call flow, you must start with a Start block, followed by any number of Middle blocks, and lastly end with an End block. For more details, refer to Hoiio IVR Documentation.

Answer

When a call is received on your Hoiio number, your server will receive a notification from Hoiio, and Hoiio expect your action - to answer the call, or ignore.

Answer is a pseudo API. You don't need to explicitly call it. To answer the call, just use a Middle or End block eg. Play a message or Transfer to another number.

The notification from Hoiio includes a *session*, which is an important data needed for you to make subsequent requests to affect that call session.

Dial

Make an outgoing to a number.

```
res = Hoiio.ivr.dial('+6511111111')

print res.txn_ref
# 'TX-1234'

print res.session
# 'S4643'
```

An advanced use which call a number with a caller ID, plays a message, and set the max duration for the call.

```
res = Hoiio.ivr.dial('+6511111111',
    msg = 'Hello. This is an automated message from Hoiio.',
    caller_id = '+6500000000',
    max_duration = '60',
    tag = 'myapp',
    notify_url = 'http://my.server.com/myscript'
)
```

Play

Play a message over the phone

```
res = Hoiio.ivr.play(session, 'Hello. This is an automated message from Hoiio.',
    tag = 'myapp',
    notify_url = 'http://my.server.com/myscript'
)

print res.is_success()
# True
```

Note: *session* represents a session of an IVR call. It is an id eg. "S-1234". You could retrieve it from Hoiio notification, or res.session after Dial.

Gather

Gather a keypad response over the phone. The following code will ask the user to press 1 for yes or press 2 for no. The system will expect only 1 digit, and it will attempt (and repeats) 3 times if the user does not respond. It will also timeout (hangup) if there is no response in 60 seconds.

```
res = Hoiio.ivr.gather(session,
    msg = 'Hello. Press 1 for yes, press 2 for no.',
    max_digits = 1,
    timeout = 60,
    attempts = 3,
    tag = 'myapp',
    notify_url = 'http://my.server.com/myscript'
)

print res.is_success()
# True
```

Record

Record a voice message.

```
res = Hoiio.ivr.record(session,
    msg = 'Hello. We are recording your voice message now.',
    max_duration = 60,
    tag = 'myapp',
    notify_url = 'http://my.server.com/myscript'
)

print res.is_success()
# True
```

Monitor

Monitor a phone conversation, that is record the whole phone conversation from the point that Monitor API is called.

```
res = Hoiio.ivr.monitor(session,
    msg = 'Hello. Note that this phone conversation is recorded.',
    tag = 'myapp',
    notify_url = 'http://my.server.com/myscript'
)

print res.is_success()
# True
```

Transfer

Transfer to a phone number or a conference room.

```
res = Hoiio.ivr.transfer(session, '+6522222222'
    msg = 'Hello. We will be transferring this call.',
    caller_id = '+6500000000',
    tag = 'myapp',
    notify_url = 'http://my.server.com/myscript'
)
```

```
print res.is_success()
# True
```

In the example code above, the call will end no matter if the transfer is successful or not. There are cases where you would want to handle the call if the transfer did not go through eg. *dest* is busy.

You could revert the Transfer operation by setting *on_failure* to 'continue'. This way, you will receive a notification when the transfer did not go through, and you can call subsequent Middle or End blocks. eg. Gather or even another Transfer.

```
res = Hoiio.ivr.transfer(session, '+6522222222'
    msg = 'Hello. We will be transferring this call.',
    caller_id = '+6500000000',
    on_failure = 'continue'
    tag = 'myapp',
    notify_url = 'http://my.server.com/myscript'
)
```

Hangup

Hangup a call.

```
res = Hoiio.ivr.monitor(session,
    msg = 'Hello. We will be hanging up now. Bye!',
    tag = 'myapp',
    notify_url = 'http://my.server.com/myscript'
)

print res.is_success()
# True
```

You can call this API at any point of time when a call is in progress. You do not need to wait for a notification before calling this API. However, if hangup is used in this way, the *msg* parameter will not be played to the user and the call will hangup immediately.

1.2.5 Fax

The Fax API can send and receive fax easily.

Send Fax

Send a PDF file to a fax number

```
res = Hoiio.fax.send('+6511111111', '/path/to/file.pdf')

print res.txn_ref
# 'TX-1234'
```

A more advanced use. The *filename* parameter is used to change the filename that resides at Hoiio.

```
res = Hoiio.fax.send('+6511111111', '/path/to/file.pdf',
    filename = 'awesome-file.pdf',
    caller_id = '+6500000000',
    tag = 'myapp',
```

```
    notify_url = 'http://my.server.com/myscript'
)
```

Receive Fax

To receive fax, you need to purchase a Hoiio number that supports receiving fax. Currently (as of Sep 2012), only Singapore numbers are capable of receiving fax.

To get a notification from Hoiio whenever you receive a fax at the number, you will need to go to Hoiio's developer portal and configure the Notify URL, or use *Number* to configure.

Retrieve Fax status

Find out the status of a Fax (supports both sent or received) with its *txn_ref*.

```
res = Hoiio.fax.status('TX-1234')

print res.txn_ref
# 'TX-1234'

print res.fax_status
# 'answered'

print res.src
# '+6500000000'

print res.dest
# '+6511111111'

print res.date
# datetime.datetime(2012, 1, 31, 12, 6, 15)

print res.fax_pages
# 3

print res.fax_url
# 'http://some.server.com/file.pdf'

print res.tag
# 'my-tag'

print res.currency
# 'SGD'

print res.rate
# 0.032

print res.debit
# 0.064
```

Retrieve Fax history

Query for all fax transactions. Each of *entry* has similar fields as the response in Retrieve Fax status.

```
res = Hoiio.fax.history()

print res.total_entries_count
# 234

print res.entries_count
# 100

for entry in res.entries:
    print entry.txn_ref
    print entry.fax_status
    # etc ..
```

You can also filter the fax history by date and type (incoming, outgoing or all).

```
res = Hoiio.fax.history(
    from = '2010-01-01 00:00:00',
    to = '2012-01-01 00:00:00',
    page = 3,
    type = 'incoming'
)
```

Retrieve Fax rate

Find out how much a fax will cost before you actually send it.

```
res = Hoiio.fax.rate('+6511111111')

print res.currency
# 'SGD'

print res.rate
# 0.032
```

You also check how much it cost to receive a fax.

```
res = Hoiio.fax.rate_in('+6500000000')

print res.currency
# 'SGD'

print res.rate
# 0.01
```

1.2.6 Number

The Number API is useful for developers to buy and manage numbers on the fly. If you need only 1 or a few numbers, you could choose to buy and manage from Hoiio developer portal.

Retrieve available countries

Retrieve the countries and states that have available Hoiio numbers. The country code and state code are needed later for Retrieve available numbers.

```
res = Hoiio.number.available_countries()

for country in res.entries:
    print '%s [%s] with number prefix %s' % (country.name, country.code, country.prefix)
    # 'USA [US] with number prefix 1'

    # For countries with states
    for state in country.states:
        print '  %s [%s]' % (state.name, state.code)
        # '  Alabama [AL]'
```

It is not necessary to use this method if the countries you support is pre-determined. You can simply use the country code in ISO 3166-1 alpha-2 format, and state code in ISO 3166-2 format.

This method is more useful if you want to support new countries dynamically as Hoiio supports them.

Retrieve available numbers

Retrieve the numbers availabe for purchasing.

```
res = Hoiio.number.available_numbers('US', 'AL')

for entry in res.entries:
    print entry.number
    # '+16001234567'

# Print the total available
print res.total_entries_count

# Print the total in this page
print len(res.entries)
```

For countries without states, the state argument can be omitted.

```
res = Hoiio.number.available_numbers('SG')
```

To access a different page,

```
res = Hoiio.number.available_numbers('SG', page=2)
```

Retrieve number cost

Retrieve how much a number subscription will cost for a country. eg. $4 per month

```
res = Hoiio.number.rate('US')

print res.currency
# 'USD'

for entry in res.entries:
    print '%d month: %f %s' % (entry.duration, entry.rate, res.currency)
    # '1 month: 4 USD'
```

Subscribe a number

Subscribe a number for *x* months.

```
# To subscribe for 1 month
res = Hoiio.number.subscribe('+16001234567', 1)

print 'Subscribed for %f %s' % (res.debit, res.currency)
print 'Expires on %s' % res.expiry
```

You can also subscribe with auto extension. That way, the number will automatically renew every month.

```
res = Hoiio.number.subscribe('+16001234567', 'auto_extend')
```

Note: Make sure you have already added your credit card in Hoiio developer portal.

Configure a number

After subscribing to a number, you can configure the number to notify your server when a call/fax/sms is received on the number.

Number capabilities varies across country; they support a mix of voice, fax and SMS, or none. Voice and fax are mutually exlusive, it's either one or the other.

As of Sept 2012:

- US numbers supports voice + SMS
- Singapore numbers supports voice/fax
- Hong Kong numbers supports voice
- Vietnam numbers supports voice
- Australia numbers supports voice
- New Zealand numbers supports voice

```
# Configure for voice only
res = Hoiio.number.configure('+16001234567',
    foward_to = 'http://my.server.com/myscript'
)

# Configure for SMS only
res = Hoiio.number.configure('+16001234567',
    foward_sms_to = 'http://my.server.com/myscript'
)

# Configure for voice + SMS
res = Hoiio.number.configure('+16001234567',
    foward_to = 'http://my.server.com/myscript',
    foward_sms_to = 'http://my.server.com/myscript',
)

# Configure for fax + SMS
res = Hoiio.number.configure('+16001234567',
    foward_to = 'http://my.server.com/myscript',
    foward_sms_to = 'http://my.server.com/myscript',
    mode = 'fax'
)
```

Retrieve subscribed numbers

Retrieve details of all your subscribed numbers.

```
res = Hoiio.number.subscribed_numbers()

for number in res.entries:
    print number.number
    # '+16001234567'

    print number.forward_to
    # 'http://my.server.com/myscript'

    print number.forward_sms_to
    # 'http://my.server.com/myscript'

    print number.expiry
    # 2012-12-31

    print number.auto_extend_status
    # 'enabled'

    print number.country
    # 'US'

    print number.state
    # 'AL'
```

1.2.7 Account

The Account API retrieves your user profile and credits balance.

Retrieve credit balance

Hoiio credit balance is made up of:

1. Main Points (aka *points*)
2. Bonus Points (aka *bonus*)

The only difference is that bonus cannot be transferred to another account. When Hoiio debits, it will take from bonus first.

```
res = Hoiio.account.balance()

print 'Balance [%s]: %f (%f + %f)' % (res.currency, res.balance, res.points, res.bonus)
# 'Balance [USD]: 12.40 (10 + 2.40)'
```

Retrieve account info

This retrieves the general information about your account.

```
res = Hoiio.account.info()

print res.uid
# 'AA-1234'
```

```
print res.name
# 'John Appleton'

print res.mobile_number
# '+16001234567'

print res.email
# 'John Appleton'

print res.country
# 'SG'

print res.currency
# 'SGD'

print res.prefix
# '1'
```

1.2.8 API Reference

A quick glance of what the SDK offers:

1. SMS
    - Hoiio.sms.send
    - Hoiio.sms.bulk_send
    - Hoiio.sms.history
    - Hoiio.sms.rate
    - Hoiio.sms.status
2. Voice
    - Hoiio.voice.call
    - Hoiio.voice.conference
    - Hoiio.voice.hangup
    - Hoiio.voice.history
    - Hoiio.voice.rate
    - Hoiio.voice.status
3. Fax
    - Hoiio.fax.send
    - Hoiio.fax.history
    - Hoiio.fax.rate
    - Hoiio.fax.status
4. IVR
    - Hoiio.ivr.dial
    - Hoiio.ivr.play

- Hoiio.ivr.gather
- Hoiio.ivr.record
- Hoiio.ivr.monitor
- Hoiio.ivr.transfer
- Hoiio.ivr.hangup

5. Number
   - Hoiio.number.available_countries
   - Hoiio.number.available_numbers
   - Hoiio.number.rate
   - Hoiio.number.subscribe
   - Hoiio.number.configure
   - Hoiio.number.active_numbers
6. Account
   - Hoiio.user.balance
   - Hoiio.user.info

1.3 Advanced

1.3.1 Clone and Setup

If you want to clone the repos (comes with docs and tests), this is what you do:

```
git clone ...
sudo pip install -r requirements.txt
sudo python setup.py install
```

You can run the test cases using nosetests. Do run the test, you need to rename *credentials-sample.py* to *credentials.py*, and enter the required credentials and details.

1.3.2 Phone Number Format

You would have noticed that all the phone numbers are in full international format eg. with + and country code and area code. However, you can change the behaviour by setting the prefix and achieve the same.

```
Hoiio.prefix = '65'
Hoiio.voice.call('11111111', '22222222')
# Equivalent to Hoiio.voice.call('+6511111111', '+6522222222')

# You could mix
Hoiio.voice.call('+8500000000', '22222222')
```

1.4 Modules & Classes

Documentation for the modules and classes available.

1.4.1 hoiio.service.voice

class hoiio.service.voice.**Voice**

Provide voice services such as making a call back, conference call, and query for calls.

Usage: Hoiio.voice.call(...), Hoiio.voice.conference(...), etc.

call(*dest1*, *dest2*, ***kwargs*)

Call 2 phone numbers and connect them up

Parameters

- **dest1** – The first phone number to call. If None, Hoiio will call the account's registered mobile number.
- **dest2** – The second phont number to call
- **caller_id** – The caller ID to show to `dest2`
- **max_duration** – Maximum duration (in seconds). Call will be hangup after `max_duration`
- **tag** – Your own reference tag for this transaction
- **notify_url** – A notification URL for Hoiio to call to your web server

Returns Return `hoiio.service.Response`

conference(**dests*, ***kwargs*)

Call multiple phone numbers and connect them up in a conference call

Parameters

- **dests** – List of phone numbers to call
- **room** (*string*) – The conference room to transfer the callers to
- **caller_id** – The caller ID to show to `dest2`
- **tag** – Your own reference tag for this transaction
- **notify_url** – A notification URL for Hoiio to call to your web server

Returns Return `hoiio.service.Response`

hangup(*txn_ref*, ***kwargs*)

Hangup a call. For conference, you need to hangup for each of the participants.

Parameters **txn_ref** – The transaction reference for the call to hangup

Returns Return `hoiio.service.Response`

history(***kwargs*)

Retrieve the history of calls. There is pagination, and each page returns up to 100 entries.

Parameters

- **from** (*string*) – Calls made after this date. In "YYYY-MM-DD HH:MM:SS" (GMT+8) format.
- **to** (*string*) – Calls made before this date. In "YYYY-MM-DD HH:MM:SS" (GMT+8) format.
- **page** (*int*) – The page number. Count starts from 1.

Returns Return `hoiio.service.Response`

rate(*dest1*, *dest2*, ***kwargs*)
Retrieve the cost of making a call

Parameters

- **dest1** – The first phone number to call.
- **dest2** – The second phont number to call

Returns Return `hoiio.service.Response`

status(*txn_ref*, ***kwargs*)
Retrieve the status and various information about a call

Parameters **txn_ref** – The transaction reference for the call

Returns Return `hoiio.service.Response`

1.4.2 hoiio.service.sms

class `hoiio.service.sms.`**Sms**
Provide SMS services such as sending and querying for SMS.

Usage: Hoiio.sms.send(...), Hoiio.sms.history(...), etc.

bulk_send(*msg*, **dests*, ***kwargs*)
Send SMS to multiple phone numbers (up to 1,000).

Parameters

- **dests** – List of phone numbers to send SMS to
- **msg** – The SMS message
- **sender_name** – The sender name that will be displayed to `dest`
- **tag** – Your own reference tag for this transaction
- **notify_url** – A notification URL for Hoiio to SMS to your web server

Returns Return `hoiio.service.Response`

history(***kwargs*)
Retrieve the history of SMS. There is pagination, and each page returns up to 100 entries.

Parameters

- **from** (*string*) – SMS sent/received after this date. In "YYYY-MM-DD HH:MM:SS" (GMT+8) format.
- **to** (*string*) – SMS sent/received made before this date. In "YYYY-MM-DD HH:MM:SS" (GMT+8) format.
- **page** (*int*) – The page number. Count starts from 1.

Returns Return `hoiio.service.Response`

rate(*msg*, *dest*, ***kwargs*)
Retrieve the cost of sending an SMS

Parameters

- **msg** – The SMS message
- **dest** – The phone number to send SMS to

Returns Return `hoiio.service.Response`

rate_in (*dest*, ***kwargs*)
Retrieve the cost of sending an SMS

Parameters **dest** – The phone number to receive the SMS at

Returns Return `hoiio.service.Response`

send (*msg*, *dest*, ***kwargs*)
Send a SMS to a phone

Parameters

- **msg** – The SMS message
- **dest** – The phone number to send SMS to
- **sender_name** – The sender name that will be displayed to `dest`
- **tag** – Your own reference tag for this transaction
- **notify_url** – A notification URL for Hoiio to SMS to your web server

Returns Return `hoiio.service.Response`

status (*txn_ref*, ***kwargs*)
Retrieve the status and various information about an SMS

Parameters **txn_ref** – The transaction reference for the SMS

Returns Return `hoiio.service.Response`

1.4.3 hoiio.service.ivr

class `hoiio.service.ivr.`**Ivr**
Provide IVR services such as Dial, Gather, Transfer, Hangup, etc.

dial (*dest*, ***kwargs*)
Make a call out to a phone number

Parameters

- **dest** – The phone number to call to
- **msg** – The message to play when the phone is answered
- **caller_id** – The caller ID when connected to the phone number
- **max_duration** – The maximum duration for the call, afterwhich it will hangup automatically
- **tag** – Your own reference tag for this transaction
- **notify_url** – A notification URL for Hoiio to call to your web server

Returns Return `hoiio.service.Response`

gather (*session*, ***kwargs*)
Gather a keypad response from the user on the phone

Parameters

- **session** – The IVR session
- **msg** – The message to play as you gather
- **max_digits** – The maximum number of digits to gather from

- **timeout** – The call will hangup if no response after timeout (in seconds)
- **attempts** – The number of times the message will be repeated
- **tag** – Your own reference tag for this transaction
- **notify_url** – A notification URL for Hoiio to call to your web server

Returns Return `hoiio.service.Response`

hangup(*session*, ***kwargs*)
Hangup a phone call

Parameters

- **session** – The IVR session
- **msg** – The message to play before you hangup
- **tag** – Your own reference tag for this transaction
- **notify_url** – A notification URL for Hoiio to call to your web server

Returns Return `hoiio.service.Response`

monitor(*session*, ***kwargs*)
Record the whole phone conversation from this point onwards

Parameters

- **session** – The IVR session
- **msg** – The message to play as you record
- **tag** – Your own reference tag for this transaction
- **notify_url** – A notification URL for Hoiio to call to your web server

Returns Return `hoiio.service.Response`

play(*session*, *msg*, ***kwargs*)
Play a message over the phone call

Parameters

- **session** – The IVR session
- **msg** – The message to play
- **tag** – Your own reference tag for this transaction
- **notify_url** – A notification URL for Hoiio to call to your web server

Returns Return `hoiio.service.Response`

record(*session*, ***kwargs*)
Record a voice message

Parameters

- **session** – The IVR session
- **msg** – The message to play before you record
- **max_duration** – The maximum duration (in seconds) for the recorded voice message
- **tag** – Your own reference tag for this transaction
- **notify_url** – A notification URL for Hoiio to call to your web server

Returns Return `hoiio.service.Response`

transfer(*session*, *dest*, ***kwargs*)
Transfer the call to another phone number or a conference room

Parameters

- **session** – The IVR session
- **dest** – The phone number or conference room. If you are transferring to a conference room, prefix with a room: eg. room:R1234.
- **msg** – The message to play as you record
- **caller_id** – The caller ID when connected to the dest
- **tag** – Your own reference tag for this transaction
- **notify_url** – A notification URL for Hoiio to call to your web server

Returns Return `hoiio.service.Response`

1.4.4 hoiio.service.fax

class `hoiio.service.fax.`**Fax**
Provide Fax services such as sending and querying for Fax.

Usage: Hoiio.fax.send(...), Hoiio.fax.history(...), etc.

history(***kwargs*)
Retrieve the history of Fax. There is pagination, and each page returns up to 100 entries.

Parameters

- **from** (*string*) – Fax sent/received after this date. In "YYYY-MM-DD HH:MM:SS" (GMT+8) format.
- **to** (*string*) – Fax sent/received before this date. In "YYYY-MM-DD HH:MM:SS" (GMT+8) format.
- **page** (*int*) – The page number. Count starts from 1.
- **type** (*incoming, outgoing or all*) – The type of fax. Default to all.

Returns Return `hoiio.service.Response`

rate(*dest*, ***kwargs*)
Retrieve the cost of sending a fax

Parameters **dest** – The fax number to send to

Returns Return `hoiio.service.Response`

rate_in(*dest*, ***kwargs*)
Retrieve the cost of receiving a fax

Parameters **dest** – The fax number to receiving at

Returns Return `hoiio.service.Response`

send(*dest*, *filename*, ***kwargs*)
Send a fax to a fax number

Parameters

- **dest** – The fax number to send fax to

- **file** – The path to the pdf file to fax
- **caller_id** – The caller ID when connected to the fax number
- **filename** – The filename to store on Hoiio. If omitted, the filename of the file will be used.
- **tag** – Your own reference tag for this transaction
- **notify_url** – A notification URL for Hoiio to call to your web server

Returns Return `hoiio.service.Response`

status(*txn_ref*, ***kwargs*)
Retrieve the status and various information about a fax

Parameters **txn_ref** – The transaction reference for the fax

Returns Return `hoiio.service.Response`

1.4.5 hoiio.service.number

class `hoiio.service.number.`**Number**
Provide Number services such as subscribing and configuring the numbers.

available_countries(***kwargs*)
Returns the countries that have Hoiio numbers.

Returns Return `hoiio.service.Response`

available_numbers(*country*, *state=None*, ***kwargs*)
Returns the available numbers for purchase

Parameters

- **country** – The country with available numbers
- **state** – The state in the country. Required for countries with states.

Returns Return `hoiio.service.Response`

configure(*number*, ***kwargs*)
Configure a number. It is recommended to use configure_voice, configure_fax or configure_sms instead of this.

Parameters

- **number** – The number to configure
- **forward_to** – The notify_url when there is an incoming call (could be voice or fax depending on mode).
- **forward_sms_to** – The notify_url when there is an incoming SMS
- **mode** (*voice or fax*) – The mode for incoming call. It can either be voice or fax (but not both). Default to voice.

Returns Return `hoiio.service.Response`

rate(*country*, ***kwargs*)
Returns the subscription cost of the country

Parameters **country** – Subscription cost of the country

Returns Return `hoiio.service.Response`

subscribe(*number*, *duration*, ***kwargs*)
Subscribe a number

Parameters

- **number** – The number to subscribe to
- **duration** (*1, 3, 12 or auto_extend*) – The number of months to subscribe to

Returns Return `hoiio.service.Response`

subscribed_numbers(***kwargs*)
Retrieve the list of subscribed numbers

Returns Return `hoiio.service.Response`

1.4.6 hoiio.service.account

class `hoiio.service.account.`**Account**
Provide account related services such as retrieve the credit balance and account information

balance(***kwargs*)
Retrieve the credit balance

Returns Return `hoiio.service.Response`

info(***kwargs*)
Retrieve account info

Returns Return `hoiio.service.Response`

1.5 Changelog

1.5.1 Version 0.1.0

Released on: *2012-09-25*

- First release!

CHAPTER 2

Indices and tables

- *genindex*
- *modindex*
- *search*

h